

# Black-box Optimization on Multiple Simplex Constrained Blocks

Priyam Das

Received: date / Accepted: date

**Abstract** Black-box optimization of objective function of parameters belonging to simplex arises in many inference and predictive models. [1] introduced Greedy Co-ordinate Descent of Varying Step-sizes on Simplex (GCDVSS) which efficiently optimizes any black-box function whose parameters belong to a simplex. In this paper, that method has been modified and extended for the case where the set of parameters may belong to multiple simplex block of different sizes. The main principle of this algorithm is to make jumps of varying step-sizes within each simplexes simultaneously and searching for the best direction for movement. Since this algorithm is designed specially for multiple simplex blocks parameter space, the proportion of movements made within the parameter space during the update step of a iteration is relatively higher for the proposed algorithm. Starting from a single initial guess, unlike genetic algorithm or simulated annealing, requirement of parallelization for this algorithm grows linearly with the dimension of the parameter space which makes it more efficient for higher dimensional optimization problems. Comparative studies with some existing algorithms have been provided based on modified well-known benchmark functions. Upto 7 folds of improvement in computation time has been noted for using the proposed algorithm over Genetic algorithm, yielding significantly better solution in all the cases considered.

**Keywords** Genetic algorithm · global optimization · simplex · Black-box optimization

---

Priyam Das  
North Carolina State University  
E-mail: pdas@ncsu.edu

## 1 Introduction

$(k - 1)$ -simplex is a  $(k - 1)$ -dimensional polytope which is the convex hull of its  $k$  vertices. Suppose  $\{v_1, \dots, v_k\} \in \mathbb{R}^{k-1}$  are  $k$  affinely independent vertices. Suppose the convex-hull generated by these vertices is called  $\mathbb{H}$ . Then all points in the  $(k - 1)$ -simplex  $\mathbb{H}$  can be described by the set

$$S_{\mathbb{H}} = \{p_1 v_1 + \dots + p_k v_k \mid p_i \geq 0, 1 \leq i \leq k, \sum_{i=1}^k p_i = 1\}.$$

Since every vector in a simplex can be represented in an unique way as a convex combination of its extreme points (see Game Theory Maschler, Solan, Zamin, pg 924), for each point in  $S_{\mathbb{H}}$ , there exist one and only one combination of  $\mathbf{p} = \{p_1, \dots, p_k\}$  satisfying the conditions  $p_i \geq 0$  for  $i = 1, \dots, k$  and  $\sum_{i=1}^k p_i = 1$ .

Consider a case where a black-box function is needed to be optimized whose constrained parameter space is given by a collection of independent simplex blocks. Suppose there are  $B$  simplex blocks given by  $\mathbf{p}_1, \dots, \mathbf{p}_B$  where  $j$ -th simplex block  $\mathbf{p}_j = (p_{j1}, \dots, p_{jn_j})$  comes from  $\Delta^{n_j-1}$  where  $\Delta^{d-1}$  is given by

$$\Delta^{d-1} = \{(x_1, \dots, x_d) \in \mathbb{R}^d \mid x_i \geq 0, i = 1, \dots, d, \sum_{i=1}^d x_i = 1\}.$$

$n_j$  denotes the number of elements in the  $j$ -th simplex block. The objective function  $f : \Delta^{n_1-1} \times \dots \times \Delta^{n_B-1} \mapsto \mathbb{R}$  is to be minimized over its domain. In other words, the problem can be re-stated as

$$\begin{aligned} & \text{minimize} : f(\mathbf{p}_1, \dots, \mathbf{p}_B) \\ & \text{subject to} : \mathbf{p}_j \in \Delta^{n_j-1}, 0 \leq j \leq B \end{aligned} \tag{1}$$

Optimization problems based on simplex parameter space arises in many problems in the field of combinatorial geometry, Hidden Markov Models, Probability vector estimation (e.g., mixture models), B-spline modeling, Genetics (e.g., [2]) etc. Among the existing methods for constrained optimization, most of them are designed for general linear or non-linear constrained space. But there is scarcity of algorithms which are designed specially for simplex parameter space.

Most of the algorithms for constrained optimization method works fine for convex functions. Among them, ‘Interior Point (IP)’ (see [3], [4], [5], [6])) and ‘Sequential Quadratic Programming (SQP)’ (see [6], [7], [8]) algorithms are widely used. Both of these methods use gradient-based approach. But the

problem with gradient based methods is that they might be time consuming for optimizing the functions with complex structure. In case, the closed form expression of the derivative of the function is not available (or provided), taking approximation of the derivative might be erroneous if the function has a lot of spikes in a small neighborhood. Lastly, although derivative-based methods works pretty fast for constrained optimization problems, these methods are prone to get stuck at local solutions for non-convex functions.

For optimizing non-convex functions, many deterministic and stochastic methods have been developed over last few decades. Among them ‘Genetic Algorithm (GA)’ (see [9], [10], [11]) and ‘Simulated Annealing (SA)’ (see [12], [13]) are widely used nowadays. Although these methods were first developed for unconstrained global optimization, later they were extended for optimizing objective functions with constraints (see [14], [15]). During optimizing the objective function, both of these methods jumps within the parameter domain for better solutions. If our interest is to optimize a function on simplex blocks, using general constrained optimization versions of GA and SA, while looking for better solution, a good proportion of points checked by the above-mentioned algorithms are prone to be generated outside the constrained space or in the infeasible region. Thus the efficiency of the algorithm might be affected. GA does not scale well with complexity because in higher dimensional optimization problems there is often an exponential increase in the search space size ([16], page 21).

As mentioned in [17], with increasing access to high-performance modern computers and clusters, some approaches like Monte Carlo methods and Multi-start strategy have a great advantage. Although they perform well for lower dimensional problems, since their requirement of parallelization grows exponentially with dimension of the parameter space, these strategies are prone to fail solve high-dimensional optimization problems efficiently.

[1] proposed ‘Greedy Co-ordinate Descent with Varying Step-size on Simplex’ (GCDVSS) algorithm which optimizes non-convex problems efficiently when the parameter is a unit simplex-block. Since that method was designed specifically for simplex parameter space, unlike constrained GA or SA, most of the jumps performed in the update step while looking for better solution fall in the feasible region, which makes it more efficient. The number of operations required in each iteration step for GCDVSS algorithm is of the order of square of the number of parameters in the simplex block which is a significant improvement over the GA whose complexity increases exponentially with the dimension ([16]). Another advantage of using GCDVSS algorithm is during an iteration, it evolves the function at  $2m$  (where  $m$  is the number of parameters in the unit simplex-block) independent directions. Thus incorporation of parallel computing makes it even faster and the requirement of parallelization increases linearly with the dimension of the simplex block.

In this paper we extend GCDVSS algorithm for the case when parameter space has multiple unit-simplex blocks (of possibly different sizes) instead of single simplex block. In GCDVSS algorithm, at each iteration the value of the objective function is evaluated at  $2m$  sites (considering there are  $m$  variables on unit simplex  $\Delta^{m-1}$ ) in the neighborhood of the current optimal solution (see Das(2016+) for details) and the best movement is selected. Similar to GCDVSS, here also in each iteration the function value is evaluated at  $2M$  sites (assuming there are total  $M$  parameters in the multiple simplex-blocks) in the neighborhood of the current optimal solution. This algorithm is named ‘Greedy Co-ordinate Descent with Varying Step-size on Multiple Simplex’ (GCDVSMS).

## 2 Algorithm

Suppose our objective function is  $f : \Delta^{n_1-1} \times \dots \times \Delta^{n_B-1} \mapsto \mathbb{R}$  which need to be minimized over its domain. This can be re-stated as the problem given in Equation (1). The main idea of movement while looking for better solution in this algorithm is quite similar to that of GCDVSS ([1]). This algorithm consists of several *runs*. Inside each *run*, iterations are performed to look for optimal solution and based on convergence criteria 1 (CC1) (see below for details), iterations inside a *run* stop and a solution is returned at the end. The next *run* starts from the solution returned by the previous *run* and tries to improve the solution. Hence only for the first *run*, the initial guess should be provided by the user. If the solution returned by two consecutive *runs* are close enough based on another convergence criteria (named convergence criteria 2 (CC2), see below for details), the algorithm stops returning the final solution. The strategy of running several *runs* helps to jump from the local solutions to a better local solution or the global solution.

In each *run*, there are four tuning parameters *initial global step size* ( $s_{initial}$ ), *step decay rate* ( $\rho$ ), *step size threshold* ( $\phi$ ) and *sparsity threshold* ( $\lambda$ ). Like GCDVSS, except the *step decay rate*, we keep the values of all other parameters same throughout all *runs*. For the first *run*, *step decay rate* is taken to be  $\rho_1$  and for the other *runs*, it is taken equal to  $\rho_2$ . Other than these five parameters  $s_{initial}$ ,  $\rho_1$ ,  $\rho_2$ ,  $\phi$  and  $\lambda$ , the two convergence criteria CC1 and CC2 are controlled by *tol\_fun\_1* and *tol\_fun\_2* respectively. Lastly, the maximum number of iterations inside a *run* and the maximum number of allowed *runs* are fixed to be equal to *max\_iter* and *max\_runs* respectively.

As mentioned in Section 1, in the parameter space consists of  $k$  unit-simplex blocks and the  $j$ -th simplex block has  $n_j$  elements in it and is denoted by  $\mathbf{p}_j = (p_{j,1}, \dots, p_{j,n_j})$  which belongs to  $\Delta^{n_j-1}$  for  $j = 1, \dots, B$ . Hence the total number of variables is  $\sum_{j=1}^B n_j = M$ . Inside each *run* there is a parameter named *global step size*  $gs$  and  $2M$  parameters named *local step sizes* which are denoted by  $s_{j,i}^+$  and  $s_{j,i}^-$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ . The values of

these above-mentioned  $2M + 1$  parameter values evolve based of the values of the previously mentioned tuning parameters and other strategies of this algorithm. Hence, the user does not need to provide or control their values. Inside a *run*, in the first iteration the value of the *global step size* is set to be  $gs^{(1)} = s_{initial}$  ( $gs^{(h)}$  denotes the value of *global step size* in the  $h$ -th iteration in a *run*). The value of  $gs$  is kept same throughout an iteration. But, at the end of each iteration, based on a convergence criteria (call convergence criteria 3 (CC3)), its value is either kept same or divided by  $\rho$  (*step decay rate*). Hence,  $gs^{(h+1)}$  will be either equal to  $gs^{(h)}$  or  $\frac{gs^{(h)}}{\rho}$  depending of the CC3 checked at the end of  $h$ -th iteration (see below and the algorithm summary for details). At the beginning of an iteration, the values of *local step sizes*  $s_{j,i}^+$  and  $s_{j,i}^-$  are set to be equal to the value of the *global step size*  $gs$  of that iteration. For example, at the start of  $h$ -th iteration of a *run*, we set  $s_{j,i}^+, s_{j,i}^- = gs^{(h)}$ . Suppose the current value of the variable at the beginning of  $h$ -th iteration is  $\mathbf{P} = \mathbf{P}^{(h)} = (\mathbf{p}_1^{(h)}, \dots, \mathbf{p}_B^{(h)})$  where  $\mathbf{p}_j^{(h)} = (p_{j,1}^{(h)}, \dots, p_{j,n_j}^{(h)}) \in \Delta^{n_j-1}$  for  $j = 1, \dots, B$ . During the iteration step, the objective function value is evaluated at  $2M$  feasible points in the neighborhood of  $\mathbf{P}^{(h)}$ . These feasible points are obtained by making the movement from the current solution based on the local step-sizes  $s_{j,i}^+, s_{j,i}^-$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$  (see below for details). The values of these local step-sizes  $s_{j,i}^+, s_{j,i}^-$  are subjected to be updated several times (if required) within an iteration (see below for details).

The above-mentioned  $2M$  movements can be divided into  $M$  ‘positive movements’  $(j, i, +)$  and  $M$  ‘negative movements’  $(j, i, -)$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ . We call a position of a unit-simplex box to be ‘significant’ if its value is greater than the sparsity control parameter  $\lambda$ . For example, since the  $j$ -th unit-simplex block has  $n_j$  positions (or elements), it can have atleast 1 and atmost  $n_j$  ‘significant’ positions. During the  $h$ -th iteration, for performing  $(j, i, +)$ -th movement,  $p_{j,i}^{(h)}$  is incremented by  $s_{j,i}^+$  and this quantity is subtracted equally from the other ‘significant’ positions of the simplex block  $\mathbf{p}_j^{(h)}$ , keeping the values of the variables of the other simplex-blocks unchanged. Thus the sum of the values in the  $j$ -th simplex block remains constant after this operation. After this update is made, it is checked whether the updated  $\mathbf{p}_j^{(h)}$  is feasible or not. In case it is feasible, the corresponding objective function value is evaluated. If the point is not feasible,  $s_{j,i}^+$  is divided by  $\rho$  (i.e., set  $s_{j,i}^+ = \frac{s_{j,i}^+}{\rho}$ ) until a feasible point is achieved. Similarly, for performing  $(j, i, -)$ -th movement,  $p_{j,i}^{(h)}$  is decremented by  $s_{j,i}^-$  and this quantity is divide by the number of other ‘significant’ positions and added to those positions of the simplex block  $\mathbf{p}_j^{(h)}$ , keeping the values of the variables of the other simplex-blocks unchanged and keeping the sum of the values of this simplex block constant, i.e., 1. Similar to the update procedure of  $s_{j,i}^+$ ,  $s_{j,i}^-$  is also subjected to be updated within the iteration until a feasible point is yielded. Thus, inside each iteration, the objective function value is evaluated at  $2M$  feasible points in the neighborhood of current solution. Only the best possible movement out of

these  $2M$  possible movements is performed at the end of the iteration which explains the greedy nature of this proposed algorithm. To control sparsity, the ‘insignificant’ positions of the simplex blocks of the finally accepted movement are set equal to 0 and the sum of the values of the ‘insignificant’ positions in a simplex block is equally divided by the number of ‘significant’ positions inside that block and added to those positions. Thus while controlling the sparsity, the sum of the elements inside every simplex block of the finally accepted movement is kept constant, i.e., 1. The minimum allowable value for  $s_{j,i}^+$  and  $s_{j,i}^-$  is  $\phi$  (*step size threshold*). At the end of an iteration, the values of the  $gs$  (*global step size*) might be changed or kept same based on convergence criteria 3 (CC3) (see below for details). The minimum allowable value of  $gs$  is  $\phi$  (*step size threshold*). At the beginning of the next iteration, the values of the *local step sizes* are set equal to the *global step size* of the next iteration.

As mentioned earlier, the value of  $gs$  is changed or kept unchanged based on CC3. CC3 is described as if the difference of the value of the objective function after an iteration with the value at the previous iteration is less than  $tol\_fun\_1$ ,  $gs$  is divided by  $\rho$ . Otherwise, it is kept unchanged. CC1 determines the stopping criteria of iterations in a *run*. CC1 is given by if at the end of an iteration CC3 is satisfied and  $gs \leq \rho * \phi$ , the iterations stop inside that *run* and the solution returned by the last iteration of that *run* serves as the starting point of the next *run*. CC2 determines the final stopping criteria of the *runs* which is if the difference of the objective function value returned by a *run* with that returned by the previous *run* is less than  $tol\_fun\_2$ , no further *run* is performed and the final solution is returned.

Below, the proposed algorithm has been noted dividing into two parts named STAGE 1 and STAGE 2. STAGE 1 describes the iterations within a *run* and STAGE 2 describes the changes of tuning parameter values and other steps performed while moving from one *run* to the next *run*. At the beginning, set  $R = 1$  ( $R$  denotes the number of *runs*),  $\rho = \rho_1$  and initial guess of the solution  $\mathbf{P}^{(1)} = (\mathbf{p}_1^{(1)}, \dots, \mathbf{p}_B^{(1)})$  where  $\mathbf{p}_j^{(1)} = (p_{j,1}^{(1)}, \dots, p_{j,n_j}^{(1)}) \in \Delta^{n_j-1}$  for  $j = 1, \dots, B$ .

#### STAGE : 1

1. Set  $h = 1$ . Set  $gs^{(h)} = s_{initial}$  Go to step (2).
2. Set  $s_{j,i}^+ = s_{j,i}^- = gs^{(h)}$  and  $f_{j,i}^+ = f_{j,i}^- = f(\mathbf{P}^{(h)}) = \mathbf{Y}^{(h)}$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ . Set  $j = 1$  and go to step (3).
3. If  $j > B$ , go to step (7). Else go to step (4).
4. If  $i > n_j$ , set  $i = 1$  and go to step (5). Else, find  $K_{j,i}^+ = n(S_{j,i}^+)$  where  $S_{j,i}^+ = \{l \mid p_{j,l}^{(h)} > \lambda, l \in \{1, \dots, n_j\} \setminus \{i\}\}$ . If  $K_{j,i}^+ \geq 1$ , go to step (4.1), else set  $i = i + 1$  and go to step (4).

- (a) If  $s_{j,i}^+ \leq \rho * \phi$ , set  $i = i + 1$  and go to step (4). Else (if  $s_{j,i}^+ > \phi$ ), evaluate vector  $\mathbf{q}_j^{i+} = (q_{j,1}^{i+}, \dots, q_{j,n_j}^{i+})$  such that

$$\begin{aligned} q_{j,l}^{i+} &= p_{j,l}^{(h)} + s_{j,i}^+ \quad \text{for } l = i \\ &= p_{j,l}^{(h)} - \frac{s_{j,i}^+}{K_{j,i}^+} \quad \text{if } l \in S_{j,i}^+ \\ &= p_{j,l}^{(h)} \quad \text{if } l \in \{1, \dots, n_j\} \setminus (S_{j,i}^+ \cup \{i\}) \end{aligned}$$

Go to step (4.2).

- (b) Check whether  $\mathbf{q}_j^{i+} \in \Delta^{n_j-1}$  or not. If  $\mathbf{q}_j^{i+} \in \Delta^{n_j-1}$ , go to step (4.3).

Else, set  $s_{j,i}^+ = \frac{s_{j,i}^+}{\rho}$  and go to step (4.1)

- (c) Evaluate  $f_{j,i}^+ = f(\mathbf{p}_1^{(h)}, \dots, \mathbf{p}_{i-1}^{(h)}, \mathbf{q}_j^{i+}, \mathbf{p}_{i+1}^{(h)}, \dots, \mathbf{p}_B^{(h)})$ . Set  $i = i + 1$  and go to step (4).

5. If  $i > n_j$ , set  $i = 1$  and go to step (6). Else, find  $K_{j,i}^- = n(S_{j,i}^-)$  where  $S_{j,i}^- = \{l \mid p_{j,l}^{(h)} > \lambda, l \in \{1, \dots, n_j\} \setminus \{i\}\}$ . If  $K_{j,i}^- \geq 1$ , go to step (5.1), else set  $i = i + 1$  go to step (5).

- (a) If  $s_{j,i}^- \leq \rho * \phi$ , set  $i = i + 1$  and go to step (5). Else (if  $s_{j,i}^- > \rho * \phi$ ), evaluate vector  $\mathbf{q}_j^{i-} = (q_{j,1}^{i-}, \dots, q_{j,n_j}^{i-})$  such that

$$\begin{aligned} q_{j,l}^{i-} &= p_{j,l}^{(h)} - s_{j,i}^- \quad \text{for } l = i \\ &= p_{j,l}^{(h)} + \frac{s_{j,i}^-}{K_{j,i}^-} \quad \text{if } l \in S_{j,i}^- \\ &= p_{j,l}^{(h)} \quad \text{if } l \in \{1, \dots, n_j\} \setminus (S_{j,i}^- \cup \{i\}) \end{aligned}$$

Go to step (5.2)

- (b) Check whether  $\mathbf{q}_j^{i-} \in \Delta^{n_j-1}$  or not. If  $\mathbf{q}_j^{i-} \in \Delta^{n_j-1}$ , go to step (5.3).

Else, set  $s_{j,i}^- = \frac{s_{j,i}^-}{\rho}$  and go to step (5.1)

- (c) Evaluate  $f_{j,i}^- = f(\mathbf{p}_1^{(h)}, \dots, \mathbf{p}_{i-1}^{(h)}, \mathbf{q}_j^{i-}, \mathbf{p}_{i+1}^{(h)}, \dots, \mathbf{p}_B^{(h)})$ . Set  $i = i + 1$  and go to step (5).

6. Set  $k_j^+ = \arg \min_{1 \leq l \leq n_j} f_{j,l}^+$  and  $k_j^- = \arg \min_{1 \leq l \leq n_j} f_{j,l}^-$ . If  $\min(f_{j,k_j^+}^+, f_{j,k_j^-}^-) < \mathbf{Y}^{(h)} (= f(\mathbf{P}^{(h)}))$ , go to step (6.1). Else, set  $\mathbf{p}_j^{temp} = \mathbf{p}_j^{(h)}$  and go to step (6.2).

- (a) If  $f_{j,k_j^+}^+ < f_{j,k_j^-}^-$ , set  $\mathbf{p}_j^{temp} = \mathbf{q}_{k_j^+}^+$ , else (if  $f_{j,k_j^+}^+ \geq f_{j,k_j^-}^-$ ), set  $\mathbf{p}_j^{temp} = \mathbf{q}_{k_j^-}^-$ . Go to step (6.2).

- (b) Define  $\mathbf{P}_j^{temp} = (\mathbf{p}_1^{(h)}, \dots, \mathbf{p}_{j-1}^{(h)}, \mathbf{p}_j^{temp}, \mathbf{p}_{j+1}^{(h)}, \dots, \mathbf{p}_B^{(h)})$  and set  $f_j^{temp} = f(\mathbf{P}_j^{temp})$ . Set  $j = j + 1$  and go to step (3).

7. Set  $w = \arg \min_{1 \leq k \leq B} f_k^{temp}$ . Set  $\mathbf{u} = \mathbf{p}_w^{temp} \in \Delta^{n_w-1}$ . Go to step (8).

8. Find  $K_{updated} = n(S_{updated})$  where  $S_{updated} = \{l \mid \mathbf{u}(l) > \lambda, l = 1, \dots, n_w\}$ . Go to step (8.1) ( $\mathbf{u}(l)$  denotes the value at the  $l$ -th co-ordinate of  $\mathbf{u}$ ).

- (a) If  $K_{updated} = n_w$ , set  $\mathbf{p}^{(h+1)} = \mathbf{u}$  and go to step (9). Else (if  $K_{updated} < n_w$ ) go to step (8.2)  
 (b) Set  $garbage = \sum_{l \in \{1, \dots, n_w\} \setminus S_{updated}} \mathbf{u}(l)$ .

$$\begin{aligned} \mathbf{u}(l) &= \mathbf{u}(l) + garbage/K_{updated} && \text{if } l \in S_{updated} \\ &= 0 && \text{if } l \in \{1, \dots, n_w\} \setminus S_{updated} \end{aligned}$$

Go to step (9).

9. Set  $\mathbf{P}^{(h+1)} = (\mathbf{p}_1^{(h)}, \dots, \mathbf{p}_{w-1}^{(h)}, \mathbf{u}, \mathbf{p}_{w+1}^{(h)}, \dots, \mathbf{p}_B^{(h)})$  and  $\mathbf{Y}^{(h+1)} = f(\mathbf{P}^{(h+1)})$ .  
 If  $(\mathbf{Y}^{(h)} - \mathbf{Y}^{(h+1)}) > tol\_fun\_1$ , set  $gs^{(h+1)} = gs^{(h)}$  and go to step (11).  
 Else (if  $(\mathbf{Y}^{(h)} - \mathbf{Y}^{(h+1)}) \leq tol\_fun\_1$ ) go to step (10).
10. If  $gs^{(h)} > \rho * \phi$  set  $gs^{(h+1)} = \frac{gs^{(h)}}{\rho}$ . Go to step (11). Else go to step (12).
11. If  $h + 1 > max\_iter$ , go to step (12). Else set  $h = h + 1$  and go to step (2).
12. **STOP** execution. Set  $\mathbf{z}^{(R)} = \mathbf{P}^{(h)}$ . Set  $R = R + 1$ . Go to STAGE 2.

### STAGE : 2

1. If  $R = 2$ , set  $\rho = \rho_2$  keeping other tuning parameters ( $\phi, \lambda$  and  $s_{initial}$ ) fixed. Repeat algorithm described in STAGE 1 setting  $\mathbf{P}^{(1)} = \mathbf{z}^{(R)}$ . Else, go to step (2) (of STAGE 2).
2. If  $R \leq max\_runs$  and  $(f(\mathbf{z}^{(R)}) - f(\mathbf{z}^{(R-1)})) \geq tol\_fun\_2$ , repeat the algorithm described in STAGE 1 setting  $\mathbf{P}^{(1)} = \mathbf{z}^{(R)}$ . Else  $\mathbf{z}^{(R)}$  is the final solution. **STOP** and **EXIT**.

The default values of the parameters are  $s_{initial} = 1$ ,  $\rho_1 = \rho_2 = 1.01$ ,  $\phi = 10^{-4}$  and  $\lambda = 10^{-6}$ . The values of  $max\_iter$  and  $max\_runs$  are taken to be 5000 and 200 respectively. It should be noted that taking lower values of  $\phi$  and  $\lambda$  results in more precise solution in the cost of higher computation time. The default values of  $tol\_fun\_1$  and  $tol\_fun\_2$  are taken to be  $10^{-6}$ .

### 3 Sparsity Control

In Section 2, note that at every iteration, after the best move is selected in a greedy manner out of  $2m$  possible moves in the neighborhood, all the values of the selected (or updated) unit-simplex block which are less than  $\lambda$  are replaced by zeros with some following adjustment (see Step (8) of STAGE 1 in Section 2). Hence if there is prior knowledge of sparsity in the final solution, the value of  $\lambda$  should be taken bigger than its default value. Thus introduction of this sparsity control parameter  $\lambda$  in this proposed algorithm helps in using the prior knowledge of sparse solution while solving the optimization problem.

### 4 Theoretical Properties

The greatest challenge of solving a non-convex optimization problem is no algorithm can be designed which always reach the global minimum while optimizing it. However, it is a desirable property of any algorithm that it should



reach a global minimum when the function is convex. In this section it has been shown that under some basic regularity conditions, taking the values of the parameters  $\phi$ ,  $tol\_fun\_1$  and  $tol\_fun\_2$  significantly small, the stopping criteria of the proposed algorithm ensures that the solution obtained is a global minimum in case the objective function is convex.

**Theorem 1** Suppose  $\mathbf{S} = \Delta^{n_1-1} \times \dots \times \Delta^{n_B-1}$  and  $f$  is convex, continuous and differentiable on  $\mathbf{S}$ . Suppose  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_B) \in \mathbf{S}$  and  $\mathbf{u}_j = (u_{j,1}, \dots, u_{j,n_j}) \in \Delta^{n_j-1}$  for  $j = 1, \dots, B$  and each of its co-ordinates are non-zero. Consider a sequence  $\delta_{j,k} = \frac{s_j}{\rho^k}$  for  $k \in \mathbb{N}$ ,  $s_j > 0$ ,  $\rho > 1$  for all  $j = 1, \dots, B$ . Define  $\mathbf{u}_{j,k}^{(i+)} = (u_{j,1} - \frac{\delta_{j,k}}{n-1}, \dots, u_{j,i-1} - \frac{\delta_{j,k}}{n-1}, u_{j,i} + \delta_{j,k}, u_{j,i+1} - \frac{\delta_{j,k}}{n-1}, \dots, u_{j,n_j} - \frac{\delta_{j,k}}{n-1})$  and  $\mathbf{u}_{j,k}^{(i-)} = (u_{j,1} + \frac{\delta_{j,k}}{n-1}, \dots, u_{j,i-1} + \frac{\delta_{j,k}}{n-1}, u_{j,i} - \delta_{j,k}, u_{j,i+1} + \frac{\delta_{j,k}}{n-1}, \dots, u_{j,n_j} + \frac{\delta_{j,k}}{n-1})$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ . If for all  $k \in \mathbb{N}$ ,  $f(\mathbf{U}) \leq f(\mathbf{u}_1, \dots, \mathbf{u}_{j-1}, \mathbf{u}_{j,k}^{(i+)}, \mathbf{u}_{j+1}, \dots, \mathbf{u}_B)$  and  $f(\mathbf{U}) \leq f(\mathbf{u}_1, \dots, \mathbf{u}_{j-1}, \mathbf{u}_{j,k}^{(i-)}, \mathbf{u}_{j+1}, \dots, \mathbf{u}_B)$  (whenever  $\mathbf{u}_{j,k}^{(i+)}, \mathbf{u}_{j,k}^{(i-)} \in \Delta^{n_k-1}$ ) for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ ,  $\mathbf{U}$  is a point of global minimum of  $f$ .

*Proof* For all  $j = 1, \dots, B$ , define,

$$S_j^* = \{(x_{j,1}, \dots, x_{j,n_j-1}) \in \mathbb{R}^{n_j-1} \mid \sum_{i=1}^{n_j-1} x_{j,i} \leq 1, x_{j,i} \geq 0, i = 1, \dots, n_j - 1\}.$$

Define  $f^* : S_1^* \times \dots \times S_B^* \mapsto \mathbb{R}$  such that  $f^*(\mathbf{x}_1^*, \dots, \mathbf{x}_B^*) = f(\mathbf{x}_1, \dots, \mathbf{x}_B)$  where  $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,n_j}) \in \Delta^{n_j-1}$  and  $\mathbf{x}_j^* = (x_{j,1}, \dots, x_{j,n_j-1}) \in S_j^*$  for  $j = 1, \dots, B$ . Note that  $\mathbf{x}_j^*$  is the first  $(n_j - 1)$  co-ordinates of  $\mathbf{x}_j$ . Consider the map  $I_j : \Delta^{n_j-1} \mapsto S_j^*$  such that  $I_j(\mathbf{x}_j) = \mathbf{x}_j^*$ . It can be seen that  $I_j$  is a bijection for any  $j = 1, \dots, B$ .

Define  $\mathbf{I} : \Delta^{n_1-1} \times \dots \times \Delta^{n_B-1} \mapsto S_1^* \times \dots \times S_B^*$  such that

$$\mathbf{I}(\mathbf{x}_1, \dots, \mathbf{x}_B) = (I_1(\mathbf{x}_1), \dots, I_B(\mathbf{x}_B)) = (\mathbf{x}_1^*, \dots, \mathbf{x}_B^*).$$

Since  $\mathbf{I}$  is the Cartesian product of bijective functions  $I_1, \dots, I_B$ ,  $\mathbf{I}$  is also a bijection. Hence, to prove that  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_B)$  is the global minimum of  $f$  on  $\Delta^{n_1-1} \times \dots \times \Delta^{n_B-1}$ , it is enough to show that  $(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*)$  is the global minimum of  $f(\mathbf{I}^{-1}(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*))$  on  $S_1^* \times \dots \times S_B^*$ . The definition of  $f^*$  reveals that  $f^* = f \circ \mathbf{I}^{-1}$ . Hence it will be sufficient to show that  $(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*)$  is a global minimum of  $f^*$  on  $S_1^* \times \dots \times S_B^*$ .

The convexity of  $f^*$  follows from the convexity of  $f$ . Consider any two points  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_B)$  and  $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_B)$  in  $\Delta^{n_1-1} \times \dots \times \Delta^{n_B-1}$ . Define  $\mathbf{X}^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_B^*)$  and  $\mathbf{Y}^* = (\mathbf{y}_1^*, \dots, \mathbf{y}_B^*)$  where  $\mathbf{x}_j^*$  and  $\mathbf{y}_j^*$  denotes the first  $(n_j - 1)$  co-ordinates of  $\mathbf{x}_j$  and  $\mathbf{y}_j$  respectively for  $j = 1, \dots, B$ . Take

any constant  $\gamma \in (0, 1)$ . Now

$$\begin{aligned}
 \gamma f^*(\mathbf{X}^*) + (1 - \gamma)f^*(\mathbf{Y}^*) &= \gamma f(\mathbf{I}^{-1}(\mathbf{X}^*)) + (1 - \gamma)f(\mathbf{I}^{-1}(\mathbf{Y}^*)) \\
 &= \gamma f(\mathbf{X}) + (1 - \gamma)f(\mathbf{Y}) \\
 &\geq f(\gamma \mathbf{X} + (1 - \gamma)\mathbf{Y}) \\
 &= f(\gamma \mathbf{I}^{-1}(\mathbf{X}^*) + (1 - \gamma)\mathbf{I}^{-1}(\mathbf{Y}^*)) \\
 &= f(\mathbf{I}^{-1}(\gamma \mathbf{X}^* + (1 - \gamma)\mathbf{Y}^*)) \\
 &= f^*(\gamma \mathbf{X}^* + (1 - \gamma)\mathbf{Y}^*)
 \end{aligned}$$

Hence  $f^*$  is convex.

Fix any  $j \in \{1, \dots, B\}$ . Define  $f_j : \Delta^{n_j-1} \mapsto \mathbb{R}$  such that

$$f_j(\mathbf{v}_j) = f(\mathbf{u}_1, \dots, \mathbf{u}_{j-1}, \mathbf{v}_j, \mathbf{u}_{j+1}, \dots, \mathbf{u}_B)$$

where  $\mathbf{v}_j = (v_{j,1}, \dots, v_{j,n_j}) \in \Delta^{n_j-1}$  for  $j = 1, \dots, B$ . Note that  $(x_{j,1}, \dots, x_{j,n_j}) \in \Delta^{n_j-1}$  implies  $(x_{j,1}, \dots, x_{j,n_j-1}) \in S_j^*$ . Define  $\mathbf{u}_j^* = (u_{j,1}, \dots, u_{j,n_j-1})$  and

$$\begin{aligned}
 \mathbf{u}_{j,k}^{*(i+)} &= (u_{j,1} - \frac{\delta_{j,k}}{n-1}, \dots, u_{j,i-1} - \frac{\delta_{j,k}}{n-1}, u_{j,i} + \delta_{j,k}, u_{j,i+1} - \frac{\delta_{j,k}}{n-1}, \dots, u_{j,n_j-1} - \frac{\delta_{j,k}}{n-1}), \\
 \mathbf{u}_{j,k}^{*(i-)} &= (u_{j,1} + \frac{\delta_{j,k}}{n-1}, \dots, u_{j,i-1} + \frac{\delta_{j,k}}{n-1}, u_{j,i} - \delta_{j,k}, u_{j,i+1} + \frac{\delta_{j,k}}{n-1}, \dots, u_{j,n_j-1} + \frac{\delta_{j,k}}{n-1}).
 \end{aligned}$$

Note that  $\mathbf{u}_j^*, \mathbf{u}_{j,k}^{*(i+)}$  and  $\mathbf{u}_{j,k}^{*(i-)}$  are the first  $n_j - 1$  co-ordinates of  $\mathbf{u}_j, \mathbf{u}_{j,k}^{(i+)}$  and  $\mathbf{u}_{j,k}^{(i-)}$  respectively.

Following the argument in the proof of Theorem 4.1 in [1], under the given conditions, there exists a positive integer  $N_j$  such that for all  $k \geq N_j$ ,  $\mathbf{u}_{j,k}^{(i+)}, \mathbf{u}_{j,k}^{(i-)} \in \Delta^{n_j-1}$ . Hence for  $k \geq N_j$ ,  $\mathbf{u}_{j,k}^{*(i+)}, \mathbf{u}_{j,k}^{*(i-)} \in S_j^*$ . Define  $f_j^* : S_j^* \mapsto \mathbb{R}$  such that

$$f_j^*(x_{j,1}, \dots, x_{j,n_j-1}) = f_j(x_{j,1}, \dots, x_{j,n_j-1}, 1 - \sum_{i=1}^{n_j-1} x_{j,i}).$$

Hence we have  $f_j^*(\mathbf{u}_j^*) = f_j(\mathbf{u}_j)$ ,  $f_j^*(\mathbf{u}_{j,k}^{*(i+)}) = f_j(\mathbf{u}_{j,k}^{(i+)})$  and  $f_j^*(\mathbf{u}_{j,k}^{*(i-)}) = f_j(\mathbf{u}_{j,k}^{(i-)})$  for  $i = 1, \dots, n_j - 1$ . Define  $\nabla_{j,i} = \frac{\partial}{\partial x_{j,i}} f_j^*(\mathbf{u}_j^*)$  for  $i = 1, \dots, n_j - 1$ . Again following the arguments made in the proof of Theorem 4.1 in [1], under the given conditions,  $\nabla_{j,i} = 0$  for  $i = 1, \dots, n_j - 1$ . Now,

$$f_j^*(\mathbf{u}_j^*) = f_j(\mathbf{u}_j) = f(\mathbf{u}_1, \dots, \mathbf{u}_B) = f(\mathbf{I}^{-1}(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*)) = f^*(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*).$$

Hence for each  $j = 1, \dots, B$ ,

$$\frac{\partial}{\partial x_{j,i}} f^*(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*) = \frac{\partial}{\partial x_{j,i}} f_j^*(\mathbf{u}_j^*) = \nabla_{j,i} = 0$$

for  $i = 1, \dots, n_j - 1$ . That implies the partial derivative of  $f^*$  with respect to each co-ordinate is zero at  $(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*)$ . Since  $f^*$  has been shown to be convex,  $(\mathbf{u}_1^*, \dots, \mathbf{u}_B^*)$  is a global minimum of  $f^*$ . Hence,  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_B)$  is a global minimum of  $f$ .

In Section 2, it should be noted that taking *step size threshold*  $\phi$  small enough, the allowable values of *local step sizes*  $s_{j,i}^+$  and  $s_{j,i}^-$  can taken as close to zero as required. Also note that in Theorem 1, the role of  $\delta_{j,k}$  is analogous to that of  $s_{j,i}^+$  and  $s_{j,i}^-$  in Section 2. In other words, in the proposed algorithm if we take *tol\_fun\_1* = 0 and *max\_iter* =  $\infty$ , the iterations within a *run* stops when for very small value of  $s_{j,i}^+$  and  $s_{j,i}^-$  for  $j = 1, \dots, B$  and  $i = 1, \dots, n_j$ , corresponding movements (as described in step (4) and (5) of STAGE 1 in Section 2) in the neighborhood do not yield better solution than the current solution. Hence, in that scenario, the obtained solution by the proposed algorithm is a global minimum if the objective function is convex with the desired minimal regularity conditions as described in Theorem 1. Note that, for a convex function satisfying the regularity conditions, the convergence criteria ensures that at the end of any *run* the solution obtained is a global minimum. Hence, in this case, evaluation of only one *run* will be enough to find the global minimum.

## 5 Simulation Studies

For simulation purpose, some multidimensional benchmark functions have been considered on transformed unit-simplex blocks parameter space. The transformations made on the parameter spaces of these benchmark functions are similar to that considered in [1]. Suppose a function  $f$  is to be minimized on a  $d$ -dimensional hypercube  $D^d$  where  $D = [l, u]$  for some constants  $l, u$  in  $\mathbb{R}$ . Consider the map  $g : D \mapsto [0, \frac{1}{d}]$  such that  $g(x_i) = y_i = \frac{x_i - l}{d(u - l)}$  for  $i = 1, \dots, d$ . Clearly  $g$  is a bijection. After replacing the original parameters of the problem with the transformed parameters we get

$$f(x_1, \dots, x_d) = f(g^{-1}(y_1), \dots, g^{-1}(y_d)),$$

where  $g^{-1}(y_i) = (u - l)dy_i + l$  for  $i = 1, \dots, d$ . Define  $h : [0, \frac{1}{d}]^d \mapsto \mathbb{R}$  such that

$$h(\mathbf{y}) = h(y_1, \dots, y_d) = f(g^{-1}(y_1), \dots, g^{-1}(y_d)).$$

Consider the set  $S = \{(z_1, \dots, z_d) \mid z_i \geq 0, \sum_{i=1}^d z_i \leq 1\}$ . Note that  $[0, \frac{1}{d}]^d \subset S$ . Define  $h' : S \mapsto \mathbb{R}$  which is equal to function  $h$  considered on the extended domain  $S$ . Since  $y_i \in [0, \frac{1}{d}]$  for  $i = 1, \dots, d$  and  $0 \leq \sum_{i=1}^d y_i \leq 1$ , hence  $0 \leq 1 - \sum_{i=1}^d y_i \leq 1$ . Define  $y_{d+1} = 1 - \sum_{i=1}^d y_i$ . Hence we can conclude that  $\bar{\mathbf{y}} = [\mathbf{y}, y_{d+1}] \in \Delta^d$  where  $\mathbf{y} = (y_1, \dots, y_d)$  and

$$\Delta^d = \{(y_1, \dots, y_{d+1}) \in \mathbb{R}^{d+1} \mid y_i \geq 0, i = 1, \dots, d+1, \sum_{i=1}^{d+1} y_i = 1\}.$$

Now define  $\bar{h} : \Delta^d \mapsto \mathbb{R}$  such that  $\bar{h}(\bar{\mathbf{y}}) = \bar{h}(y_1, \dots, y_{d+1}) = h'(y_1, \dots, y_d)$  for  $\bar{\mathbf{y}} \in \Delta^d$ . It can be seen that  $\bar{\mathbf{y}} \in \Delta^d$  implies  $(y_1, \dots, y_d) \in S$ . Suppose the global minimum of the function  $f$  occurs at  $(m_1, \dots, m_d)$  in  $D^d$ . Hence, the function  $\bar{h}$  will have the global minimum at  $\bar{\mathbf{y}} = (g^{-1}(m_1), \dots, g^{-1}(m_d), 1 - \sum_{i=1}^d g^{-1}(m_i))$  in  $\Delta^d$ .

Now consider there are  $n$  blocks of simplexes  $\bar{\mathbf{y}}_r \in \Delta^d$  for  $r = 1, \dots, n$ . Define,

$$H(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) = \bar{h}(\bar{\mathbf{y}}_1) + \dots + \bar{h}(\bar{\mathbf{y}}_n)$$

Here the comparative study of performances of the proposed algorithm and various existing methods of constrained optimization has been shown for optimizing some benchmark unconstrained global optimization problems on transformed parameter space which is given by  $n$  blocks of  $d$ -dimensional simplexes. Among the other methods we considered the ‘interior-point’ (IP) algorithm, ‘sequential quadratic programming’ (SQP) and constrained ‘genetic algorithm’ (GA). IP and SQP search for local minimum while optimizing any function and in general they are less time consuming. On the other hand GA tries to find global minimum being more time consuming. These above-mentioned well-known algorithms are available in Matlab R2014a (The Mathworks) via the Optimization Toolbox functions *fmincon* (for IP and SQP algorithm) and *ga* (for GA). For GCDVSMS algorithm, the values of all the tuning parameters have been taken to be same as mentioned in Section 2. While using IP and SQP algorithms, the upper bound for maximum number of iterations and function evaluations have been set to be infinity each. For GA, the default options of ‘ga’ function in Matlab R2014a has been considered. GCDVSMS algorithm is implemented in Matlab R2014a. The comparative study has been performed for the cases  $n = 5, d = 5$  and  $n = 5, d = 10$  for all the above-mentioned algorithms. To check the performance of the proposed algorithm in higher dimensional problems, additional simulation studies have been performed. For each cases, all the algorithms have been initialized from 100 randomly generated starting points. The average time (in seconds) and minimum value of the objective function for each cases have been noted down in Table I. All the computations have been performed in a computer with 64-Bit Windows 8.1, Intel i7 3.6GHz processor, 32GB RAM.

### 5.1 Modified Rastrigin Function

$d$ -dimensional Rastrigin function is given by

$$f(\mathbf{x}) = 10d + \sum_{i=1}^d [x_i^2 - \cos(2\pi x_i)], \quad \mathbf{x} = (x_1, \dots, x_d) \in D \quad (2)$$

where  $D$ , the domain of the parameters are typically taken to be  $[-5.12, 5.12]^d$ . Hence we have  $l = -5.12, u = 5.12$ . After performing the above-mentioned transformations, we obtain

$$\bar{h}(\bar{\mathbf{y}}) = 10d + \sum_{i=1}^d [(10.24dy_i - 5.12)^2 - \cos(2\pi(10.24dy_i - 5.12))], \bar{\mathbf{y}} = (y_1, \dots, y_{d+1}) \in \Delta^d. \quad (3)$$

We consider the case where we need to minimize  $H(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) = \sum_{v=1}^n \bar{h}(\bar{\mathbf{y}}_v)$  for  $\bar{\mathbf{y}}_v \in \Delta^d$  for  $v = 1, \dots, n$ . In Table I, it is noted that GCDVSMS outperformed other algorithms and the average computation time of GCDVSMS is 4-5 folds smaller than that of GA.

## 5.2 Modified Ackley's Function

$d$ -dimensional Ackley's function is given by

$$f(\mathbf{x}) = -20 \exp(-0.2 \sqrt{0.5 \sum_{i=1}^d x_i^2 - \exp(0.5(\sum_{i=1}^d \cos(2\pi x_i))})) + e + 20, \mathbf{x} = (x_1, \dots, x_d) \in D \quad (4)$$

where  $D$ , the domain of the parameters are typically taken to be  $[-5, 5]^d$ . After necessary transformations, we get

$$\bar{h}(\bar{\mathbf{y}}) = -20 \exp(-0.2 \sqrt{0.5 \sum_{i=1}^d (10dy_i - 5)^2 - \exp(0.5(\sum_{i=1}^d \cos(2\pi(10dy_i - 5))))}) + e + 20, \quad (5)$$

where  $\bar{\mathbf{y}} = (y_1, \dots, y_{d+1}) \in \Delta^d$ . Our objective is to minimize  $H(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) = \sum_{v=1}^n \bar{h}(\bar{\mathbf{y}}_v)$  over  $\bar{\mathbf{y}}_v \in \Delta^d$  for  $v = 1, \dots, n$ . In this case also, GCDVSMS outperforms all other algorithms with 2-3 fold time improvement over GA.

## 5.3 Modified Sphere Function

All of the above-mentioned functions being non-convex, we consider the Sphere function which is convex.  $d$ -dimensional Sphere function is given by

$$f(\mathbf{x}) = \sum_{i=1}^d x_i^2, \mathbf{x} = (x_1, \dots, x_d) \in D \quad (6)$$

Here the domain of the parameters  $D$  is taken to be  $[-5.12, 5.12]^d$ . The modified Sphere function on simplex is given by,

$$\bar{h}(\bar{\mathbf{y}}) = \sum_{i=1}^d (10.24dy_i - 5.12)^2, \bar{\mathbf{y}} = (y_1, \dots, y_{d+1}) \in \Delta^d. \quad (7)$$

We minimize  $H(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) = \sum_{v=1}^n \bar{h}(\bar{\mathbf{y}}_v)$  over  $\bar{\mathbf{y}}_v \in \Delta^d$  for  $v = 1, \dots, n$ . It should be noted that, the Sphere function being convex, IP and SQP functions are expected to work better than GCDVSMS and GA while minimizing it. Note that, GCDVSMS yields significantly better solution than GA with 3-4 folds improvement in computation time.

#### 5.4 Modified Griewank Function

$d$ -dimensional Griewank function is given by

$$f(\mathbf{x}) = \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right), \mathbf{x} = (x_1, \dots, x_d) \in D \quad (8)$$

Here the domain of the parameters  $D$  is taken to be  $[-500, 500]^d$ . After transformation, the Griewank function on simplex is given by,

$$\bar{h}(\bar{\mathbf{y}}) = \sum_{i=1}^d \frac{(1000dy_i - 500)^2}{4000} - \prod_{i=1}^d \cos\left(\frac{1000dy_i - 500}{\sqrt{i}}\right), \bar{\mathbf{y}} = (y_1, \dots, y_{d+1}) \in \Delta^d. \quad (9)$$

We minimize  $H(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) = \sum_{v=1}^n \bar{h}(\bar{\mathbf{y}}_v)$  over  $\bar{\mathbf{y}}_v \in \Delta^d$  for  $v = 1, \dots, n$ . For the case  $n = 5, d = 5$ , GCDVSMS performs significantly better than others. In the other case, IP and SQP performs better than GA and GCDVSMS. In this case also, GCDVSMS outperforms GA with 4-7 folds improvement in computation time.

**Table I** Comparative study of GCDVSMS, IP, SQP and GA for optimizing modified Rastrigin, Ackley, Sphere and Griewank function starting from 100 randomly generated initial points.

Functions	Algorithms	$n = 5, d = 5$		$n = 5, d = 10$	
		Min. value	Avg. time	Min. value	Avg. time
Modified Rastrigin	GCDVSMS	1.39e + 01	10.180	1.70e + 01	14.586
	IP	3.02e + 02	0.369	1.24e + 03	681.627
	SQP	1.89e + 02	0.232	5.91e + 02	0.681
	GA	1.51e + 01	52.597	2.32e + 02	56.903
Modified Ackley's	GCDVSMS	3.48e - 02	13.412	6.51e - 02	24.204
	IP	5.35e + 01	0.395	6.33e + 01	0.998
	SQP	1.18e + 01	0.254	1.17e + 01	0.674
	GA	6.63e + 00	46.834	1.60e + 01	58.566
Modified Sphere	GCDVSMS	7.42e - 05	11.676	5.13e - 04	27.534
	IP	1.30e - 16	0.224	6.01e - 15	0.598
	SQP	3.39e - 12	0.158	3.55e - 11	0.920
	GA	1.72e + 00	44.381	1.34e + 02	85.520
Modified Griewank	GCDVSMS	4.94e - 01	9.849	9.29e - 01	19.193
	IP	9.13e + 00	0.870	1.40e - 01	1.396
	SQP	10.09e + 00	0.563	7.35e - 01	1.635
	GA	10.42e + 00	72.054	2.73e + 02	81.796

**Table II** Performance of GCDVSMS in higher dimensional modified Rastrigin, Ackley, Sphere and Griewank function starting from 100 randomly generated initial points.

Dimensions	Modified Rastrigin		Modified Ackley		Modified Sphere		Modified Griewank	
	Value	Time	Value	Time	Value	Time	Value	Time
$n = 20, d = 10$	1.03e - 02	101.35	4.01e - 02	162.44	5.24e - 05	58.51	1.13e - 00	25.25
$n = 20, d = 20$	2.04e - 02	104.20	4.01e - 02	273.58	1.02e - 04	105.23	1.09e - 01	245.23
$n = 50, d = 10$	2.58e - 02	1514.48	1.00e - 01	794.19	1.30e - 04	315.46	6.43e - 00	776.93
$n = 50, d = 20$	5.07e - 02	810.42	1.00e - 01	1655.51	2.60e - 04	609.05	2.43e - 01	3750.52

## 6 Discussion

This paper proposes a black-box optimization technique where the parameter space is given by a collection of independent simplex blocks. In the comparative study provided in table I, it is noted that the proposed algorithm outperformed all the other considered existing algorithms. Also it should be noted that the proposed algorithm works upto 7 times faster than Genetic algorithm yielding better solution under each scenarios considered. In table II some other higher dimensional simulation studies have been also provided on modified Rastrigin, Ackley, Sphere and Griewank function where the proposed algorithm reaches significantly close to the true solution in reasonable time.

**Acknowledgements** I would personally like to thank my Ph.D. advisor Dr. Subhashis Ghoshal who motivated me into this work as the major part of my Ph.D. project. I would also like to thank Dr. Hua Zhou who enriched my knowledge regarding optimization and efficient coding techniques.

## References

1. Das, P., Derivative-free efficient global optimization on high-dimensional simplex, arXiv:1604.08636 (2016)
2. Blei, D.M. and Lafferty, J.D., Topic models, <http://www.cs.columbia.edu/blei/papers/BleiLaerty2009.pdf> (2009)
3. Potra, F.A. and Wright, S.J., Interior-point methods. Journal of Computational and Applied Mathematics, 4, 281-302 (2000)
4. Karmakar, N., New polynomial-time algorithm for linear programming, COMBINATORICA, 4, 373-395 (1984)
5. Boyd, S. and Vandenberghe, L., Convex Optimization. Cambridge: Cambridge University Press (2006)
6. Wright, M. H., The interior-point revolution in optimization: History, recent developments, and lasting consequences, Bulletin of American Mathematical Society, 42, 39-5 (2005)
7. Nocedal, J. and Wright, S.J., Numerical Optimization, 2nd edition., Operations Research Series, Springer, (2006)
8. Boggs, P.T. and Tolle, J.W., Sequential quadratic programming, Acta Numerica 4, 1-51 (1996)
9. Fraser, A.S., Simulation of genetic systems by automatic digital computers i. introduction. Australian Journal of Biological Sciences, 10, 484-491 (1957)
10. Bethke, A.D., Genetic algorithms as function optimizers, (1980)
11. Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Operations Research Series. Addison-Wesley Publishing Company (1989)
12. Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., Optimization by simulated annealing. Australian Journal of Biological Sciences, 220, 671-680, (1983)

- 
13. Granville, V., Krivanek, M. and Rasson, J.P., Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 652-656 (1994)
  14. Smith, R.L. and Romeijn, H.E., Simulated annealing for constrained global optimization, *Journal of global optimization, Journal of Global Optimization*, 5(2), 101-126 (1994)
  15. Reid, D.J., Genetic algorithms in constrained optimization, *Mathematical and Computer Modelling*, 23(5), 87-111 (1996)
  16. Geris, L., *Computational Modeling in Tissue Engineering*, Springer (2012)
  17. Hilbert, M. and Lopez, P., The worlds technological capacity to store, communicate, and compute information, *Science*, 332, 60-65 (2011)